A Mobility Model Incorporating Obstacle Avoidance for Evaluation of Proactive Scheduling Algorithms in the mmWave Band

Ang Deng
Georgia Institute of Technology
June 2023

CREATING THE NEXT®

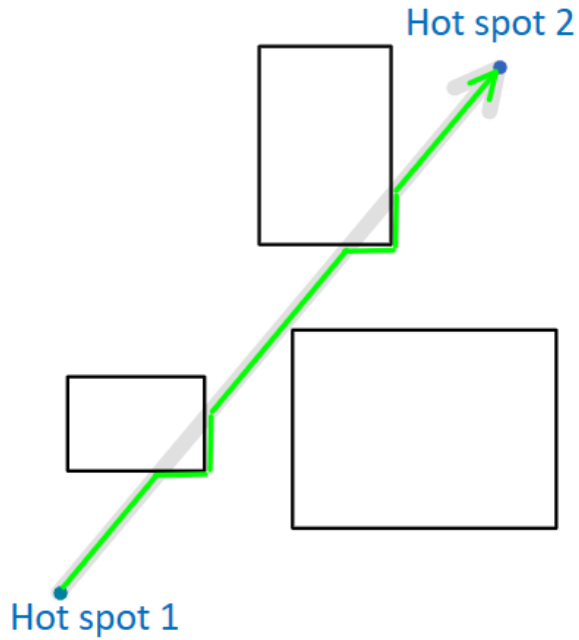
Motivation

- Why mmWave?
  - Modern applications are increasingly bandwidth demanding
- Why proactive scheduling?
  - mmWave sensitivity to blockage
    - Penetration loss up to 30dB
- Blockage duration can be up to 300ms, while applications are delay-sensitive
- Our prior work has shown that proactive scheduling can achieve a 30% increase in aggregate rate compared to classic proportional fair scheduling (PFS) with no decrease in fairness

Mobility Model

- Internally developed hot spot mobility model with obstacle avoidance:
  - Hot spots at which users pause for some time after which they either stay or move to a different hotspot
  - When moving between hot spots, users take the shortest route while circumventing any obstacles in between

black: obstacles
gray: direct path
green: path with obstacle avoidance



Implementation

- Mobility models in ns-3 works by scheduling next node activities (next function) after the duration of current activity
- A basic hotspot mobility model would rotate between walk() and pause() function.

```
walk():
    next = pick next hotspot;
    node.update(next);
    duration = calculate duration according to
               current and target location;
    Schedule(duration, pause);
```

```
pause():
    node.pause();
    duration = pause duration;
    Schedule(duration, walk);
```

* the implementation is based on the WiGig module release , on version ns-3.31.

Implementation

- In our implementation, first, the obstacle model needs to be incorporated into the mobility model
- For obstacle avoidance, after picking the next hot spot, the model calculates the direct path, and then checks if there is any interceptions with obstacles.
  - If so, circumventing paths are planned

Implementation

```
beginWalk():
    next = pick next hotspot;
    intercept = list of intercepting obstacles;
    targets = series of target locations forming
              path segments from current location to
              target location;
    index = 0;
    node.update(targets[index]);
    index = index + 1;
    duration = calculate duration according to
               current and target location;
    if index == targets.size()
        Schedule(duration, pause);
    else
        Schedule(duration, contWalk);
```

```
contWalk():
    node.update(targets[index]);
    index = index + 1;
    duration = calculate duration according to
               current and target location;
    if index == targets.size()
        Schedule(duration, pause);
    else
        Schedule(duration, contWalk);
```

```
pause():
    node.pause();
    duration = pause duration;
    Schedule(duration, beginWalk);
```

Future Work

- Future work will integrate a proactive scheduling algorithm into ns-3 to demonstrate the performance benefits of proactive scheduling with mobility prediction.

Georgia Tech

Thank you!